Scope, Closures

Scope

Scope: where to look for things

Understanding Scope

A scope in JavaScript defines what variables you have access to. There are two kinds of scope – **global scope** and **local scope**.

Global scope

If a variable is declared outside all functions or curly braces ({}), it is said to be defined in the global scope.

```
const hello = 'Hello Reader!'

function sayHello () {
  console.log(hello)
}

console.log(hello)
sayHello()
```

Local Scope

Variables that are usable only in a specific part of your code are considered to be in a local scope. These variables are also called local variables.

Function Scope

When you declare a variable in a function, you can access this variable only within the function.

```
function person() {
   var name = "Maria";
   console.log(name);
}
console.log(name);
```

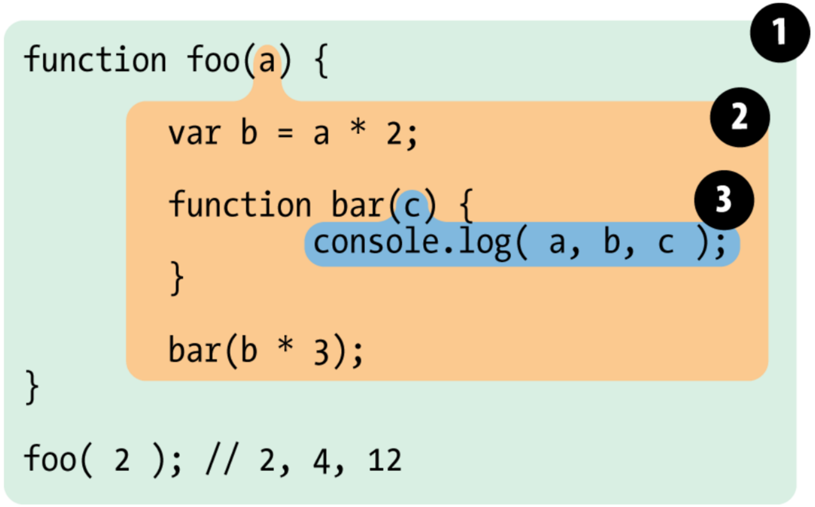
Block Scope

When you declare a variable with const or let within a curly brace ({}), you can access this variable only within that curly brace.

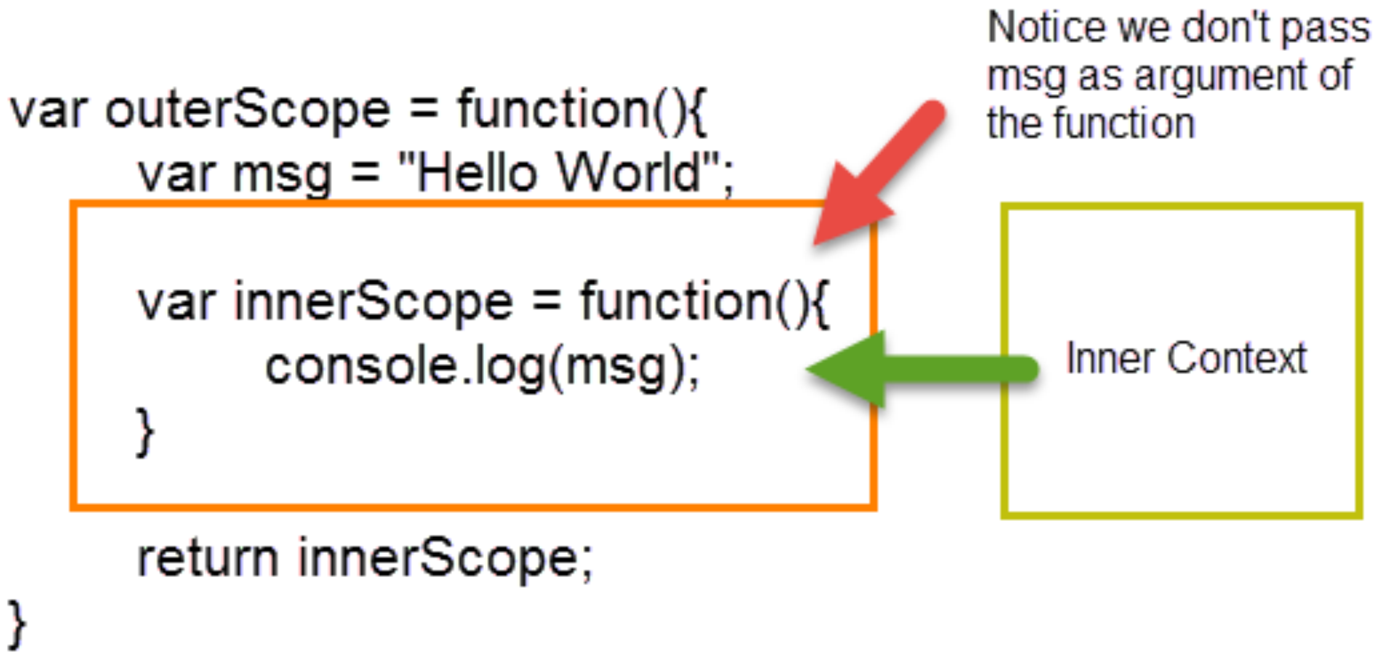
```
{
    let stars = null;
    stars = getStars();
    console.log( stars );
}
console.log( stars );

for (let i=0; i<10; i++) {
    console.log( i );
}
```

Lexical Scope

When a function is defined in another function, the inner function has access to the outer function's variables.



```
function foo(a) {
        var b = a * 2;
        function bar(c) {
                console.log( a, b, c );
        }
        bar(b * 3);
}
foo( 2 ); // 2, 4, 12
```



```
var outerScope = function(){
        var msg = "Hello World";

        var innerScope = function(){
                console.log(msg);
        }

        return innerScope;
}
```

Hoisting

Hoisting

variable and function declarations are physically moved to the top of your code

Variable hoisting

Only declarations are hoisted

```
console.log(num); // Returns undefined
var num; // Declaration
num = 6; // Initialization

// ==========

console.log(num); // Throws ReferenceError
num = 6; // Initialization

// ==========

a = 1; // initialization.
let a; // Throws SyntaxError
```

Function hoisting

A function declaration are always hoisted to the top of the current scope

```
sayHello()

function sayHello () {
    console.log('Hello!')
}

sayHello()
```

Functions First

Both function declarations and variable declarations are hoisted but functions are hoisted first, and then variables.

```
sayHello()
var sayHello = "Hello"

function sayHello () {
   console.log('Hello!')
}

sayHello()
```

Closures

What is Closure

Closure is when a function "remembers" its lexical scope even when the function is executing outside that lexical scope.

```
function counter(step = 1) {
  var count = 0;
  return function increaseCount() {
    count = count + step;
    return count;
  };
}

var incBy1 = counter(1);
var incBy3 = counter(3);

incBy1();
incBy1();

incBy3();
incBy3();
incBy3();
```

Private variables with closures

```
function secret (secretCode) {
   return {
       saySecretCode () {
           console.log(secretCode)
       }
   }
}
const theSecret = secret('JS is amazing')
theSecret.saySecretCode()
```

Controlling side effects with closures

```
function loadData(url) {
 return function () {
   return fetch(url).toJson();
 };
}

const loadDataLater = loadData("https://...");

renderTableHead();
const data = loadDataLater()
renderTableBody(data)
```

Operators

What are Operators

Operators are ”symbols” that are used to operate with the data we’re dealing with

Arithmetic

```
+ sum
- difference
/ division
* multiplication
++ increment
-- decrement
```

Logic

```
&& - and
|| - or
! - negation
```

What are Operators

Operators are ”symbols” that are used to operate with the data we’re dealing with

Comparisons

```
===
!=
==
!==
>=
<=
```

What are Operators

Operators are ”symbols” that are used to operate with the data we’re dealing with

Attribution

```
+
-
*
/
+=
-=
/=
*=
```

Flow control

Flow control

Flow control, debugger

```
if / else

for loop

while / do while

switch
```

Quiz

Quiz

https://create.kahoot.it/creator/72c97c8e-7243-45b0-ab08-a940aabb937e

Assignment

Assignment - 1

Go through exercises JS Variables – JS Functions from W3 Schools

https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_variables1

Assignment - 2

- In a script.js file create a Student Class (commit files to a folder called assignment-C6)
- A student instance should have the following properties:
    - name (String)
    - phone_number (String)
    - age (Number)
    - hobbies (Array)
- Name and Phone number are mandatory when creating the class instance (in constructor)
- Hobbies will have a separate setter/getter method which will be called after class instance is created

Assignment - 2

- Using a “for…” loop iterate through an array of students e.g.
  ```
  ["Sharleen Rollo", "Della Wally", "Ryana Ami", "Lydia Mercy",
  "Mikey Valorie", "Chester Lexie", "Danette Luanna", "Adalyn
  Goddard", "Johnnie Peta", "Natille Nigellus"]
  ```
- and create a Student instance with ***all*** the student properties (yes, including hobbies)
- console.log that student instance (each student instance)
- At least two of the created Student should have age 25, and at least three should have hobbies like ’music’ or ‘books’
- Student class should have a method called `greetings`, which when invoked should return a salute like “Hello, my name is Sharleen Rollo and I’m 24 of age.” (where Sharleen and age number are dynamic of course)

Assignment – 2 (extra mile 1/2)

- After you've done this, we see that our Student class objects are only available within loop, and that's not so cool. We want to use them after we created them
- Declare an array variable (e.g. allStudents) before iterating through the for loop
- allStudents variable should be initialized with an empty array.
- Inside the for loop complete the loop so that the empty array would populate with the Student class objects (use something like `allStudents[allStudents.length] = myValue;`
- Avoid array push for now (we will come back to it later down our journey)
- console.log allStudents

Assignment – 2 (extra mile 2/2)

- Now iterate through allStudents and console.log the ones that have hobbies like `music` or `books` but not just simply console.log them, but console log their greeting message
- (So the console.log should be like this:

“Hello, my name is Sharleen Rollo and I’m 24 of age.” – but only for students that have the mentioned hobbies)